SUBSTITUTE SPECIFICATION

METHOD FOR THE AUTOMATIC RETRIEVAL OF ENGINEERING DATA FROM INSTALLATIONS

[0001] This application is the national phase under 35 U.S.C. § 371 of PCT International Application No. PCT/DE00/00735 which has an International filing date of March 9, 2000, which designated the United States of America, the entire contents of which are hereby incorporated by reference.

Field of the Invention

[0002] The invention relates to a method for the automatic retrieval of engineering data from installations.

Background of the Invention

[0003] An automation system of this type is used in particular in the area of automation technology. An automation system of this type generally comprises a multiplicity of individual automation objects, which are frequently highly dependent on the engineering system respectively used.

[0004] At present there are two basic methods in use. In the first method, the retrieval of the engineering data from the installation is ruled out. Changes to the installation are possible only via the engineering tool. Consequently, the data in the engineering system always reflect the current state and there is no need for information to be reproduced from the installation. This solution has the following disadvantages:

Strong link between runtime and engineering: The engineering system must be supplied along with the installation and also be additionally paid for by the customer.

Changes in the installation cannot be reproduced: If there are changes in the installation, for example as a result of a device being exchanged, these changes cannot be automatically reproduced in the engineering system.

High organizational expenditure: To keep the engineering data up to date, organizational precautions have to be taken to ensure the way in which changes in the installation are introduced into the engineering system.

[0005] The second approach is based on a disassembly of the runtime code. In this case, the executable code of the runtime objects is analyzed and translated into the engineering counterparts. This solution has the following disadvantages:

- Elaborate method: The analysis of the runtime code is complex and susceptible to errors.
- Implementation-dependent: The implementation of the translation back is strongly dependent on how the translation process is carried out. Changes to the translation process and in particular the code created necessitate adaptation of the implementation of the translating-back process.
- ES information can no longer be produced with certainty: Since the runtime code is at a semantically lower level than the actual engineering information, it cannot be ensured that the engineering information can be exactly reconstructed.

[0006] In the specialist article Elmqvist, H.: "A Uniform Architecture for Distributed Automation", Advances in Instrumentation and Control, vol. 46, part 2, 1991, pages 1599-1608, XP000347589 Research Triangle Park, NC, US, a description is given of an automation system whose objects are programmed in an object- and data-flow-oriented programming language. It uses a graphic programming environment and offers means for the creation of dynamically updated process images. The programming language allows an automatic communication between distributed objects.

SUMMARY OF THE INVENTION

[0007] One problem underlying the invention is that of allowing the information contained in an installation to be automatically reproduced in an engineering system and used again there, for example to plan changes in the installation.

[0008] An object of the invention is to solve that and/or other problems by a method and by a system with the features specified in claims 1 and 8, respectively.

[0009] In this case, the engineering and runtime objects are described by a uniform object model. As a result, the correspondence between engineering objects and runtime objects can be determined at the object level and no information is lost as a result of the mapping. In addition, a direct communication between engineering and runtime objects can take place, which can be utilized when the method is carried out.

[0010] The relationship between an engineering object and its runtime counterpart is described in figure 1. The engineering object ESO has a direct reference, RTO ref, to its runtime counterpart RTO. This is possible since the runtime objects are available (or become available) at the time of engineering. The runtime object RTO has no direct reference to the associated engineering object. This is necessary to make it possible for the engineering system and runtime system to be separated. Instead of this, the object RTO contains an identifying designation, ESO type ID, referring to the type of engineering object, ESO type. Consequently, required instances of the ESO type can then be created by the RTO.

[0011] With respect to a runtime object RTO, the method for the restoration of engineering information proceeds as follows:

1. If a runtime object receives the order to retrieve its engineering information, it firstly addresses the type of its engineering object with the order to create a new instance of an engineering object.
2. In the newly created instance, the runtime object enters a reference to itself and orders the new engineering object to read out its data (that of the runtime object).
3. The new engineering object then reads out the information from the runtime object and enters the corresponding engineering information in itself.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The invention is described and explained in more detail below on the basis of the exemplary embodiments represented in the figures, in which:

Figure 1 shows an overview to identify the relationships between engineering objects and runtime objects,
Figure 2 shows a view of an object of an installation by way of example,
Figure 3 shows an illustration of the creation of device representatives in the engineering,
Figure 4 shows a representation of the creation of the automation objects in the device representatives by way of example and
Figure 5 shows a layout of the existing communication relationships in the engineering.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0013] The method for the retrieval of engineering information from the installation preferably proceeds in three steps:

Restoration of the device representatives;
Restoration of the representatives of the automation objects in the engineering; and
Restoration of the communication relationships between the representatives of the automation objects.

[0014] The method is described below for the complete retrieval of the engineering information. However, it can equally be used for updating already existing engineering information, i.e. as a delta method. Hereafter, the overall method is referred to as upload.

[0015] In figure 2, the objects involved are listed by way of example. Two automation objects run on each of the two devices RG1 and RG2. The automation objects RAO1 and RAO2 run on RG1, RAO3 and RAO4 run on RG2. Communication connections are symbolized by lines. Thus, altogether two device-internal and two device-interlinking communication relationships exist.

1. Restoration of the device representatives

[0016] The beginning of the upload is initiated from a software system. This may be an engineering system or any other desired system which requires engineering information. One example of this is a system for parameterizing the installation. For the sake of simplicity, hereafter reference is always made to an engineering system.

[0017] In the first step, all the devices are requested to create their representation in the engineering. For this purpose, each device returns an identifier of the type of its engineering counterpart. The engineering system then creates the corresponding objects and enters the reference to the actual device in each device representative. By means of the reference, each device representative then reads out the relevant data of "its" device.

[0018] Figure 3 illustrates what has just been described. The devices of the installation, here RG1 and RG2, receive the request to upload through the engineering system. They then in each case return the identifiers of the types of the engineering representatives. The engineering system creates the instances G1 and G2 for the corresponding types. These then read out the relevant engineering information from the devices RG1 and RG2.

2. Restoration of the automation objects in the engineering

[0019] In the second step, the representatives of the automation objects are created in the engineering. Via the device assigned to it, each device representative requests the automation objects of its device to create its counterparts in the engineering. For this purpose, each automation object returns the identifier of the type of its engineering representative. In the engineering system, the corresponding objects are then again created and provided with a reference to their partner in the runtime environment. After that, each automation object in the engineering inquires the relevant data of its partner.

[0020] The result of this operation can be seen in figure 4. The representative G1 inquires from the device RG1 the automation objects RAO1 and RAO2. These are then requested to upload by G1 and return the identifiers of the types of AO1 and AO2. By means of this information, the instances AO1 and AO2 are created in the engineering. These then receive a reference to their runtime counterparts RAO1 and RAO2 are finally assigned to the device representative G1. As a result, the information on the device assignment of the automation objects is available again. Subsequently, AO1 and AO2 read out the information relevant for engineering from RAO1 and RAO2.

3. Restoration of the communication relationships between the automation objects in the engineering

[0021] In the third step, the communication relationships between the automation objects are restored. For this purpose, each device representative asks the device assigned to it for its communication relationships. The device then returns a list with both the device-internal and device-interlinking communication relationships. An entry of this list comprises the source and drain of the communication relationship. The source and drain are in each case described by a 3-tuple from the identifier of the physical device, the identifier of the automation object and the identifier of the input or output.

[0022] In the engineering system, the entries of the list are converted into references to the inputs and outputs of the representatives of the automation objects. For this purpose, the information from the already created objects (the references of the engineering representatives to their runtime counterparts) is used. Subsequently, the connection in the engineering system is then set up.

[0023] An efficient way of carrying out the step will ensure that the list with communication connections created by each device only contains those in which the device appears in the identifier of the source (alternatively of the drain). Furthermore, an effective method will buffer-store the relationships between engineering representatives and runtime counterparts set up in steps 1 and 2, in order in this way to minimize the searching effort in step 3.

[0024] Figure 5 then shows the result of the last step. G1 has inquired the communication relationships from RG1. In response, the relationships between RAO1 and RAO2, RAO1 and RAO3 and between RAO2 and RAO4 were returned. The connections are then converted in the engineering, for example the connection between RAO1 and RAO3 is converted to the connection between AO1 and AO3.

[0025] Both the objects of the engineering system and of the runtime system are based on the same, executable object model. The use of the same model makes possible a direct interaction at model level (data exchange and communication) between the engineering objects and runtime objects. Furthermore, a unique mapping, which is independent of the implementation of the objects, is defined by the defined assignment between the engineering and runtime objects.

[0026] This gives rise to advantages for the method, including but not limited to:

> **Separation of engineering and runtime possible:** Changes do not necessarily have to be carried out with the engineering tool. If need be, the changes can be introduced into the engineering system at any time.

Simple method: By determining the method at the level of explicit models, the method can be described in general terms and so becomes more reliable.

Simple and complete mapping: There is a defined relationship between the runtime and engineering objects, making complete restoration of the engineering information possible.

Stable with respect to changes in implementation: Implementation of the runtime and engineering objects can be changed over without having any influence on the mapping and consequently on the way in which the method is carried out.

Non-tool-specific: The upload mechanism can also be used by other tools and not just by the engineering system.

[0027] The invention being thus described, it will be obvious that the same may be varied in many ways. Such variations are not to be regarded as a departure from the spirit and scope of the invention, and all such modifications as would be obvious to one skilled in the art are intended to be included within the scope of the following claims.